Documentação do Sistema de Análise Preditiva com Agentes de IA

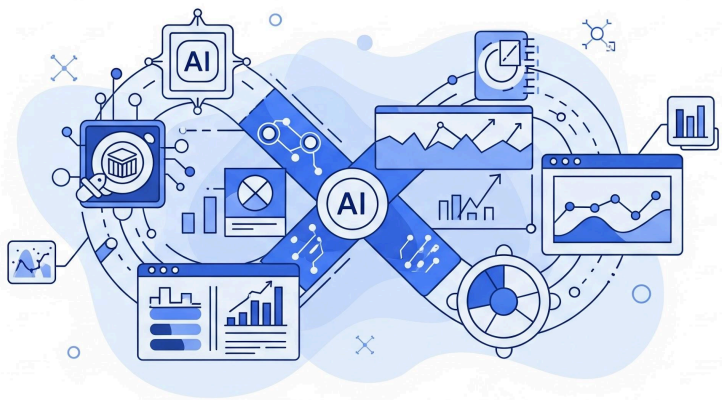
Este documento detalha a arquitetura, componentes e funcionalidades do sistema de análise preditiva baseado em múltiplos agentes de IA. Ele foi projetado para servir como um guia abrangente para desenvolvedores e stakeholders técnicos, facilitando a compreensão e a implementação do sistema.

1. Visão Geral e Objetivos

O Sistema de Análise Preditiva com Agentes de IA é uma solução inovadora projetada para automatizar e aprimorar o processo de análise de dados e previsão de resultados de negócio. Utilizando um conjunto de agentes de inteligência artificial especializados, o sistema é capaz de receber contexto de negócio, integrar-se a diversas bases de dados, aplicar modelos preditivos avançados e gerar dashboards interativos com visualizações de dados dinâmicas.

Os principais objetivos do sistema incluem:

- **Automação da Análise Preditiva:** Automatizar a ingestão de dados, pré-processamento, aplicação de modelos e geração de insights.
- **Decisão Baseada em Dados:** Fornecer previsões precisas e visualizações claras para apoiar a tomada de decisões estratégicas.
- **Flexibilidade e Escalabilidade:** Permitir a fácil adaptação a diferentes contextos de negócio e escalar horizontalmente para lidar com volumes crescentes de dados.
- **User Experience Aprimorada:** Oferecer uma interface web intuitiva com dashboards interativos para exploração de dados e insights.



2. Arquitetura do Sistema

O sistema é construído sobre uma arquitetura baseada em agentes de IA, com um pipeline de dados automatizado e uma forte ênfase em modularidade e escalabilidade.

2.1. Visão Geral da Arquitetura

O diagrama a seguir ilustra a arquitetura geral do sistema:

```
None
graph TD

    A[Usuário/Sistema Externo] -->|Requisição| B(API REST)
|---|
    C --> D[Data Processing Agent]
    C --> E[Context Analysis Agent]
    C --> F[Predictive Analysis Agent]
    C --> G[Visualization Agent]
    C --> H[Dashboard Agent]
    D -- Dados Processados --> F
    E -- Contexto Interpretado --> F
    F -- Predições --> G
    G -- Visualizações --> H
    H -->|Dashboards Interativos| I(Frontend Web)
```

```
    I -->|API REST| B
    D -- Conexão com DBs --> J[Bases de Dados]
    K[Docker/Kubernetes] -- Deploy --> L[Ambiente de Produção]
```

2.2. Componentes Principais

- **API REST (FastAPI/SpringBoot):** Ponto de entrada para todas as interações externas, permitindo a ingestão de contexto de negócio e a recuperação de resultados.
- **Agentes de IA (Python):** Módulos especializados com responsabilidades distintas, comunicando-se entre si para executar o fluxo de análise.
- **Pipeline de Dados Automatizado:** Garante a ingestão, processamento e transformação contínua dos dados.
- **Frontend Web (React):** Aplicação web responsiva para visualização de dashboards interativos.
- **Containerização (Docker/Kubernetes):** Facilita o deploy, gerenciamento e escalabilidade dos componentes do sistema.

3. Documentação Detalhada dos Agentes

Cada agente de IA possui uma responsabilidade específica no sistema. Abaixo, detalhamos cada um, incluindo um exemplo de sua estrutura de código Python.

3.1. Agent Coordinator

Orquestra todo o processo de análise, coordenando a execução dos demais agentes.

Python
```python
# agents/coordinator_agent.py
class AgentCoordinator:
    def __init__(self):
        # Inicialização de outros agentes
        self.data_processing_agent = DataProcessingAgent()
        self.context_analysis_agent = ContextAnalysisAgent()
        self.predictive_analysis_agent = PredictiveAnalysisAgent()
        self.visualization_agent = VisualizationAgent()
        self.dashboard_agent = DashboardAgent()

    def run_analysis(self, business_context, data_sources):
```

```
        # 1. Processar dados
        processed_data = self.data_processing_agent.process(data_sources)
        # 2. Analisar contexto
        interpreted_context =
self.context_analysis_agent.analyze(business_context)
        # 3. Gerar predições
        predictions = self.predictive_analysis_agent.predict(processed_data,
interpreted_context)
        # 4. Criar visualizações
        visualizations = self.visualization_agent.create_visuals(predictions)
        # 5. Gerar dashboard
        dashboard_output =
self.dashboard_agent.organize_dashboard(visualizations, predictions)
        return dashboard_output
```

3.2. Data Processing Agent

Responsável pela limpeza, normalização e pré-processamento dos dados brutos.

Python

```python
# agents/data_processing_agent.py
import pandas as pd

class DataProcessingAgent:
    def process(self, data_sources):
        # Exemplo: Conectar a DB e carregar dados
        # data = load_from_database(data_sources['db_config'])
        # Exemplo simulado
        data = pd.DataFrame({
            'feature1': [10, 20, 30, 40, 50],
            'feature2': [100, 200, 150, 250, 300],
            'target': [15, 25, 35, 45, 55]
        })
        # Limpeza e normalização
        processed_data = self._clean_and_normalize(data)
        return processed_data

    def _clean_and_normalize(self, df):
        # Lógica de limpeza (e.g., tratar NAs, remover duplicatas)
        df = df.dropna()
        # Lógica de normalização (e.g., StandardScaler)
        # scaler = StandardScaler()
```

```
        # df[['feature1', 'feature2']] = scaler.fit_transform(df[['feature1',
'feature2']])
        return df
```

3.3. Context Analysis Agent

Interpreta o contexto de negócio fornecido para influenciar a análise preditiva.

Python
```python
# agents/context_analysis_agent.py
class ContextAnalysisAgent:
    def analyze(self, business_context):
        # Lógica para interpretar o contexto (e.g., identificar
variáveis-chave, períodos)
        interpreted_context = {
            "key_variables": ["feature1", "feature2"],
            "prediction_period": "next_quarter",
            "business_segment": business_context.get("segment", "general")
        }
        return interpreted_context
```

3.4. Predictive Analysis Agent

Executa a regressão linear e gera as predições com base nos dados processados e contexto.

Python
```python
# agents/predictive_analysis_agent.py
from sklearn.linear_model import LinearRegression
import joblib

class PredictiveAnalysisAgent:
    def __init__(self):
        self.model = None

    def train_model(self, X, y):
        self.model = LinearRegression()
        self.model.fit(X, y)
        # Opcional: Salvar o modelo
```

```
        # joblib.dump(self.model, 'linear_regression_model.pkl')

    def predict(self, processed_data, interpreted_context):
        features = processed_data[interpreted_context["key_variables"]]
        target = processed_data['target'] # Assumindo que o target está aqui
para treino

        if self.model is None:
            self.train_model(features, target) # Treina se o modelo não existir

        # Para predição, normalmente você teria novos dados sem o 'target'
        # Aqui, simulamos a predição sobre os mesmos dados processados para
demonstração
        predictions = self.model.predict(features)

        # Gerar predições com base no contexto (ex: para o próximo período)
        # Este é um placeholder, a lógica real dependeria de como o contexto
influencia a predição
        future_data = features * 1.1 # Exemplo de dados
```